IN THE CLAIMS

1. (Currently Amended) A method of accessing a memory array implemented in a computer-readable medium, comprising:

providing data contained within a one-dimensional array of allocated memory;

dynamically declaring a dimensional dynamic overlay on the data contained within the one-dimensional array from within a block of statements in a ~~user-defined software program~~ subroutine to initialize attributes within an array attribute storage object, wherein the dynamic overlay is defined within the subroutine and provides a dimensional view on the one-dimensional array, the dimensional dynamic overlay being capable of providing a view of at least two dimensions on the one-dimensional array; and

accessing the data from within the block of statements as a dimensional indexed array using the array attribute storage object.

2. (Original) The method of claim 1, wherein declaring a dimensional dynamic overlay on the data includes:

providing a pointer to the one-dimensional array of allocated memory;

providing an array access identifier; and

providing array information for the declared dimensional dynamic overlay.

3. (Original) The method of claim 2, wherein providing array information includes providing array height information, array width information and array stride information.

4. (Original) The method of claim 2, further including performing allocated memory offset and array index boundary calculations on the array information that is for the dimensional dynamic overlay.

5. (Original) The method of claim 1, wherein declaring a dimensional dynamic overlay on the data contained within the one-dimensional array includes coding a dimensional dynamic overlay declaration using extended programming language from within the subroutine.

6. (Original) The method of claim 1, further comprising setting an explicit boundary policy for the declared dimensional dynamic overlay.

7. (Currently Amended) A method of accessing a memory array implemented in a computer-readable medium, comprising:

providing data contained within a one-dimensional array of allocated memory;

dynamically declaring a dimensional dynamic overlay on the data contained within the one-dimensional array from within a block of statements in a ~~user-defined software program~~ subroutine, wherein the dynamic overlay is defined within the subroutine and provides a dimensional view on the one-dimensional array, the dimensional dynamic overlay being capable of providing a view of at least two dimensions on the one-dimensional array;

providing a dynamic overlay storage object associated with the declared dimensional overlay;

assigning attributes from the declared dimensional dynamic overlay to the storage object; and

accessing the data from within the block of statements as a dimensional indexed array using the declared dimensional overlay.

8. (Original) The method of claim 7, wherein providing a dynamic overlay storage object includes providing a dynamic overlay storage object within a hardware environment.

9. (Original) The method of claim 7, wherein providing a dynamic overlay storage object includes providing a dynamic overlay storage object within a software environment.

10. (Original) The method of claim 7, further comprising automatically freeing the dynamic overlay storage object when leaving the block of statements in which the dimensional dynamic overlay was declared.

11. (Original) The method of claim 7, wherein declaring a dimensional dynamic overlay on the data includes:

providing a reference to the one-dimensional array of allocated memory;

providing an array access identifier; and

providing array information for the declared dimensional dynamic overlay.

12. (Original) The method of claim 11, wherein providing array information includes providing array height information, array width information and array stride information.

13. (Original) The method of claim 11, further including performing allocated memory offset and array index boundary calculations on the array information for the dimensional dynamic overlay.

14. (Currently Amended) A method of creating and accessing a multi-dimensional dynamic array implemented in a computer-readable medium, comprising:

dynamically declaring a dimensional dynamic array from within a block of statements in a ~~user-defined~~ software program subroutine;

dynamically allocating memory storage sufficient to store all the elements for the declared dimensional dynamic array;

providing a dynamic overlay storage object with attributes initialized from the dimensional dynamic array declaration, wherein the dynamic overlay storage object provides a dimensional view on a one-dimensional array associated with contiguous memory locations, the dimensional dynamic overlay storage object being capable of providing a view of at least two dimensions on the one-dimensional array;

accessing data from the dynamically allocated memory storage as a dimensional indexed array from within the block of statements using the dynamic overlay storage object;

automatically freeing the dynamically allocated memory storage when leaving a subroutine in which the dynamic array is declared; and

automatically freeing the dynamic overlay storage object when leaving a subroutine in which the dynamic array is declared.

15. (Original) The method of claim 14, wherein providing a dynamic overlay storage object with attributes initialized from the declared dimensional dynamic array includes:

providing a pointer to a corresponding one-dimensional array in the dynamically allocated memory storage;

providing a handle for an array access; and

providing array information.

16. (Original) The method of claim 15, further including performing allocated memory offset and array index boundary calculations on the array information provided for the declared dimensional dynamic array.

17. (Original) The method of claim 15, wherein providing array information includes providing array height information, array width information and array stride information.

18. (Original) The method of claim 14, wherein providing a dynamic overlay storage object includes providing a dynamic overlay storage object separately stored within a hardware environment to enhance performance.

19. (Original) The method of claim 14, wherein providing a dynamic overlay storage object includes providing a dynamic overlay storage object allocated within normal memory in a software environment.

20. (Original) The method of claim 14, wherein declaring a dimensional dynamic array on the data contained within the one-dimensional array includes coding a dimensional dynamic overlay declaration using extended programming language from within the subroutine.

21. (Original) The method of claim 14, further comprising setting an explicit array boundary policy for the declared dimensional dynamic array.

22. (Currently Amended) A method of processing a data array implemented in a computer-readable medium, comprising:

providing a software program with at least one block of statements;

declaring a data array within the block of statements, including a dimensional dynamic overlay to provide a dimensional view on a one-dimensional array associated with contiguous memory locations, the dimensional dynamic overlay being capable of providing a view of at least two dimensions on the one-dimensional array;

setting an [[an]] array boundary policy for the data array which is defined with or referenced by some of the block of statements, wherein the array boundary policy dictates run-time actions of the software program that are executed, if during execution of the software program, the data array is accessed outside its boundaries;

accessing the array within the block of statements;

compiling the software program; and

executing the software program.

23. (Original) The method of claim 22, wherein declaring a data array within the subroutine includes:

declaring an array access handle;

declaring an array size; and

declaring a data element type.

24. (Original) The method of claim 23, wherein declaring an array size includes:

determining the number of data array dimensions for a problem; and

selecting the total number of data array dimensions.

25. (Original) The method of claim 24, wherein selecting the total number of data array dimensions includes providing a dimension size expression that is constant.

26. (Original) The method of claim 24, wherein selecting the total number of data array dimensions includes providing a dimension size expression that is evaluated at run-time.

27. (Original) The method of claim 22, wherein executing the software program includes:

performing run-time allocation of memory to obtain a base address attribute;

performing run-time calculation of array size attributes from the declared data array;

obtaining the array boundary policy setting as an array attribute;

associating attributes of the declared data array with the array access handle; and

accessing the data array at run-time using the array access handle and array indices.

28. (Original) The method of claim 27, wherein accessing the data array at run-time using the array access handle and array indices comprises:

obtaining the attributes of the declared data array using the array access handle; and

performing run-time boundary policy enforcement based on array access attributes.

29 (Original) The method of claim 28, further comprising terminating the program if the array boundary policy aborts for an invalid index.

30. (Original) The method of claim 27, wherein accessing the data array at run-time using the array access handle and array indices comprises:

obtaining the attributes of the declared data array using the array access handle;

performing run-time invalid index value detection based on array access attributes for the declared data array;

applying the array boundary policy to constrain the invalid index values to be valid index values;

calculating an offset into the declared data array from the valid index values and the attributes of the declared data array; and

adding the offset to the base address attribute to obtain a memory address for accessing the indexed data array element.

31. (Previously Presented) The method of claim 22, wherein setting the array boundary policy further includes setting a reflect-at-boundary policy that reflect array data at a declared boundary.

32. (Previously Presented) The method of claim 22, wherein setting the array boundary policy further includes setting a confined index and boundary policy that replicates data beyond a detected boundary.

33. (Previously Presented) The method of claim 22, wherein setting the array boundary policy further includes setting a pre-defined array attribute value that is to be read for all out of bounds accesses.

34. (Currently Amended) A method of processing a data array implemented in a computer-readable medium, comprising:

providing a software program with a block of statements in at least one subroutine;

declaring a dimensional dynamic overlay on data contained within a one-dimensional array from within the block of statements to initialize attributes within an array attribute storage object, wherein the dimensional dynamic overlay provides a dimensional view on the one-dimensional array, the dimensional dynamic overlay being capable of providing a view of at least two dimensions on the one-dimensional array;

setting an array boundary policy for the declared dimensional overlay which is defined with or referenced by some of the block of statements wherein the array boundary policy dictates run-time actions of the software program that are executed, if during execution of the software program, the data array is accessed outside its boundaries;

accessing the data from within the block of statements as a dimensional indexed array using the array attribute storage object;

compiling the software program; and

executing the software program.

35. (Previously Presented) The method of claim 34, wherein setting the array boundary policy further includes setting a reflect-at-boundary policy that reflect array data at a declared boundary.

36. (Previously Presented) The method of claim 34, wherein setting the array boundary policy further includes setting a confined index and boundary policy that replicates data beyond a detected boundary.

37. (Previously Presented) The method of claim 34, wherein setting the array boundary policy further includes setting a pre-defined array attribute value that is to be read for all out of bounds accesses.

38. (Currently Amended) A method of processing a data array implemented in a computer-readable medium, comprising:

providing a software program with a block of statements in at least one subroutine;

declaring a dimensional dynamic array from within the block of statements to initialize attributes within an array attribute storage object;

dynamically allocating memory storage for the declared dimensional dynamic array;

providing a dynamic overlay storage object with attributes assigned from the declared dimensional dynamic array, wherein the overlay storage object maps to the memory storage of the declared dimensional dynamic array, the overlay storage object being capable of mapping at least two dimensions on a one-dimensional array associated with the memory storage;

setting an array boundary policy for the declared dimensional dynamic array which is defined with or referenced by some of the block of statements wherein the array boundary policy dictates run-time actions of the subroutine that are executed, if during execution of the subroutine, the data array is accessed outside its boundaries;

accessing data from the dynamically allocated memory storage as a dimensional indexed array from within the block of statements using the dynamic overlay storage object;

automatically freeing the dynamically allocated memory storage when leaving a subroutine in which the dynamic array is declared;

automatically freeing the dynamic overlay storage object when leaving a subroutine in which the dynamic array is declared;

compiling the software program; and

executing the software program.

39. (Previously Presented) The method of claim 38, wherein setting the array boundary policy further includes setting a reflect-at-boundary policy that reflect array data at a declared boundary.

40. (Previously Presented) The method of claim 38, wherein setting the array boundary policy further includes setting a confined index and boundary policy that replicates data beyond a detected boundary.

41. (Previously Presented) The method of claim 38, wherein setting the array boundary policy further includes setting a pre-defined array attribute value that is to be read for all out of bounds accesses.

42. (Currently Amended) A system, comprising:

a computer readable medium;

an extended programming language encoded in the computer readable medium, including:

dynamic array language extensions for declaring a dimensional dynamic array and creating an array handle for accessing the dimensional dynamic array; and

dynamic overlay language extensions for declaring a dimensional dynamic overlay on existing allocated data and creating an array handle for accessing the dimensional dynamic overlay, wherein the dimensional dynamic overlay provides a dimensional view on the existing allocated data, the dimensional dynamic overlay is capable of providing a view of at least two dimensions; and

a translator for converting the extended programming language implemented in ~~user-defined~~ programs into machine code instructions that are able to be run on a processor.

43. (Original) The system of claim 42, wherein the translator includes a compiler adapted for converting the extended language into machine code instructions that is able to be run on the processor.

44. (Original) The system of claim 42, wherein the translator includes:

a language converter for converting the extended language into a mid-level programming code; and

a compiler for compiling the mid-level language program code into machine code instructions that is able to be run on the processor.

45. (Original) The system of claim 44, wherein:

the language converter includes a converter program encoded on the computer readable medium; and

the processor executes the converter program to read the extended language and convert the extended language into the mid-level programming code.

46. (Original) The system of claim 42, wherein:

the extended language is a C programming language with language extensions; and

the language converter converts the extended language into C code.

47. (Original) The system of claim 42, further including at least one software program subroutine encoded in the computer readable medium, wherein the at least one software program subroutine includes the extended programming language, and wherein the at least one software program accesses data in the dimensional dynamic arrays and the dimensional dynamic overlays.

48. (Currently Amended) A system, comprising:

a computer readable medium;

an extended programming language encoded in the computer readable medium, including:

dynamic array language extensions for declaring a dimensional dynamic array and creating an array handle for accessing the dimensional dynamic array;

dynamic overlay language extensions for declaring a dimensional dynamic overlay on existing allocated data and creating an array handle for accessing the dimensional dynamic overlay, wherein the dimensional dynamic overlay provides a dimensional view on the existing allocated data, the dimensional dynamic overlay is capable of providing a view of at least two dimensions; and

boundary policy language extensions for setting array boundary policies for the dimensional dynamic array and the dimensional dynamic overlay, which define run-time actions to process when run-time access of the dimensional dynamic array exceeds its boundaries; and

a translator for converting the extended programming language into machine code instructions that is able to be run on a processor.

49. (Original) The system of claim 48, further including at least one software program subroutine encoded in the computer readable medium, wherein the at least one software program subroutine includes the extended programming language, and wherein the at least one software program accesses data in the dimensional dynamic arrays and the dimensional dynamic overlays.

50. (Original) The system of claim 48, wherein the translator includes a cross compiler.

51. (Original) The system of claim 48, wherein the translator includes a native compiler.